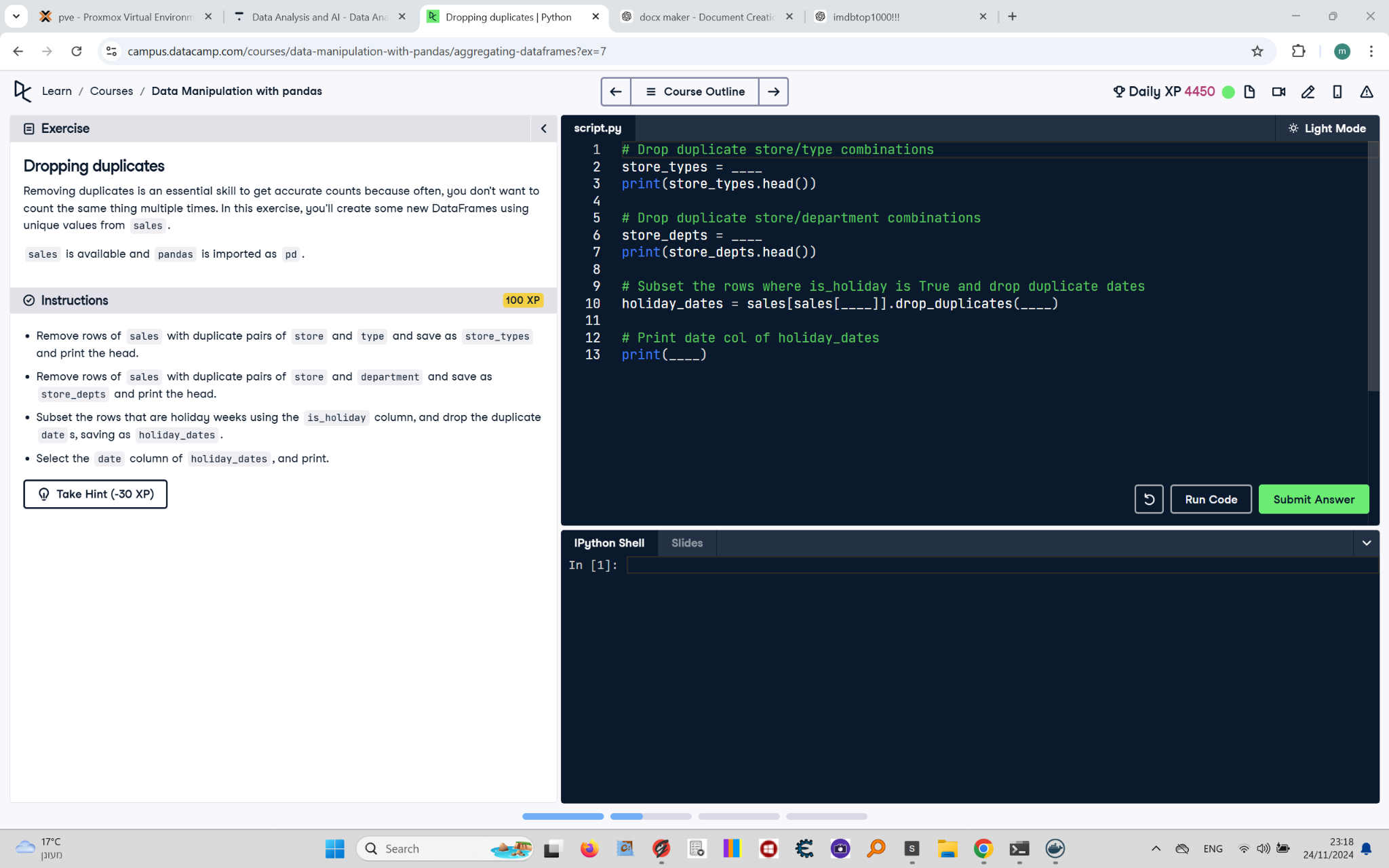
# Dropping Duplicates (Updated Solution)

This document includes the question, the updated solution, and a breakdown of the code provided in the screenshot.

## Uploaded Screenshot

Below is the updated screenshot of the task:



## Question

1. Remove rows of `sales` with duplicate pairs of `store` and `type` and save as `store\_types`, then print the head.  
2. Remove rows of `sales` with duplicate pairs of `store` and `department` and save as `store\_depts`, then print the head.  
3. Subset the rows that are holiday weeks using the `is\_holiday` column and drop duplicate `date`s, saving as `holiday\_dates`.  
4. Select the `date` column of `holiday\_dates` and print.

## Updated Answer

# Drop duplicate store/type combinations  
store\_types = sales.drop\_duplicates(subset=['store', 'type'])  
print(store\_types.head())  
  
# Drop duplicate store/department combinations  
store\_depts = sales.drop\_duplicates(subset=['store', 'department'])  
print(store\_depts.head())  
  
# Subset rows where is\_holiday is True and drop duplicate dates  
holiday\_dates = sales[sales['is\_holiday']].drop\_duplicates(subset='date')  
  
# Print date column of holiday\_dates  
print(holiday\_dates['date'])

## Code Explanation

# Explanation of the code:

1. `sales.drop\_duplicates(subset=['store', 'type'])`: Removes duplicate rows based on the combination of `store` and `type` columns.

2. `sales.drop\_duplicates(subset=['store', 'department'])`: Removes duplicate rows based on the combination of `store` and `department` columns.

3. `sales[sales['is\_holiday']]`: Filters the rows where the `is\_holiday` column is `True`.

4. `.drop\_duplicates(subset='date')`: Removes duplicate rows based on the `date` column within the filtered DataFrame.

5. `holiday\_dates['date']`: Selects the `date` column from the `holiday\_dates` DataFrame.